Predictive Text Entry for Agglutinative Languages Using Unsupervised Morphological Segmentation

Miikka Silfverberg, Krister Lindén, and Mirka Hyvärinen

University of Helsinki
Department of Modern Languages
Helsinki, Finland
{miikka.silfverberg, krister.linden, mirka.hyvarinen}@helsinki.fi

Abstract. Systems for predictive text entry on ambiguous keyboards typically rely on dictionaries with word frequencies which are used to suggest the most likely words matching user input. This approach is insufficient for agglutinative languages, where morphological phenomena increase the rate of out-of-vocabulary words. We propose a method for text entry, which circumvents the problem of out-of-vocabulary words, by replacing the dictionary with a Markov chain on morph sequences combined with a third order hidden Markov model (HMM) mapping key sequences to letter sequences and phonological constraints for pruning suggestion lists. We evaluate our method by constructing text entry systems for Finnish and Turkish and comparing our systems with published text entry systems and the text entry systems of three commercially available mobile phones. Measured using the keystrokes per character ratio (KPC) [8], we achieve superior results. For training, we use corpora, which are segmented using unsupervised morphological segmentation.

1 Introduction

Mobile phone text messages are a hugely popular means of communication, but mobile phones are not especially well-suited for inputting text because of their small size and often limited keyboard. There exist several technological solutions for text entry on mobile phones and other limited keyboard devices. This paper is concerned with a technology called *predictive text entry*, which utilizes redundancy in natural language in order to enable efficient text entry using limited keyboards (typically having 12 keys).

The subject of predictive text entry has been extensively studied, but the studies have mainly concentrated on predictive text entry of English. Because of the limited morphological complexity of English, these approaches have usually been able to rely on an extensive dictionary along with word frequencies, since a sufficiently large English dictionary almost eliminates the problem of out-of-vocabulary (OOV) words. E.g. [5] reports low OOV word rates of 1.42% for a training set containing the 40,000 most frequent words in the North American

Business News Corpus and a test set consisting of 54,265 sentences from the same corpus.

For morphologically complex languages like Finnish and Turkish, productive inflection, derivation and compounding raise the number of OOV words regardless of the size of the dictionary, i.e. the vocabulary growth rate does not converge [2]. This means that OOV words present a serious problem for dictionary based approaches to predictive text entry of languages like Finnish and Turkish.

In this paper we present an approach to predictive text entry based upon a morphologically segmented training corpus, which is used to construct a probabilistic model of morphotax. We additionally use a probabilistic model on letter sequences and two phonological constraints, which constrain the results of the probabilistic models. We show that this combination delivers superior results compared with a system based on a colloquial dictionary and a morphological analyzer [11] for text entry of Finnish, when evaluated on actual text-message data using the keystroke per character ratio (KPC) [8]. Thus we achieve superior results to [11] without using labour intensive linguistic resources such as morphological analyzers. Additionally, we compare our method to the predictive text entry in three commercially available mobile phones and show that our approach gives superior KPC.

Apart from two phonological rules, our approach is entirely unsupervised and data-driven, since we use the unsupervised morphological segmentation system Morfessor [3] for segmenting the training corpus and the tools for constructing POS-taggers from the HFST interface [7]. We show that our method can also be applied to another agglutinative language [1] besides Finnish, namely Turkish. We compare the Turkish text entry system with an existing text entry system, which is based on a Markov model on letter sequences and show that our approach gives a substantial improvement in KPC.

The paper is structured as follows. In Section 2 we present some earlier approaches to predictive text entry. In Section 3, we present the components of our model for text entry and explain how these models are combined into a system for predictive text entry. In Section 4 we describe the training and test corpora used in constructing and testing predictive text entry systems for Finnish and Turkish together with the phonological rules which are used to realize Finnish vowel harmony. Evaluation of the systems is presented in Section 5 and the results are discussed in Section 6. Finally we present some concluding remarks and future work directions in Section 7.

2 Related Approaches to Text Entry

The mobile phone keypad is a so called *clustered keyboard*, where each key can be used to enter several letters. E.g. on the Finnish mobile phone keypad in Figure 1 key "2" is used to enter the letters "a", "b", "c", "ä" and "å".

[1] Agglutinative languages are characterized by extensive use of inflectional and derivational affixes as well as compounding.

The original method for text entry is the so called *multitap-method*. When entering text in multitap mode, each key is pressed multiple times to scroll through the list of letters that are associated with the key. As text-messages have gained popularity, other faster methods for text entry have been devised. These can broadly be classified into *movement minimization techniques*, which concentrate on keypad layout, and *language prediction techniques*, which use linguistic models to disambiguate ambiguous user input [10].

The most widely used language prediction techniques are based on a dictionary. They disambiguate suggestions based on word frequencies. The best known example of a dictionary based system is the commercially successful T9-system [4]. There are many variants of dictionary based methods. E.g. some methods try to guess the word before all characters have been typed. Some approaches also include information on the probability of word sequences [10].

Fig. 1. The 12-key keypad of a typical Finnish mobile phone. There are three letters in the Finnish alphabet “ä”, “å” and “ö”, which are not shown on the keypad. The letters “ä” and “å” are entered pressing key “2” four times and five times respectively. The letter “ö” is entered by pressing key “6” four times.

As we noted in the introduction, dictionary based methods are not optimal for agglutinative languages, where the OOV rate remains high even with large dictionaries. Two alternative approaches better suited for agglutinative languages are known to the authors: prefix-based disambiguation [9] and disambiguation of output using a probabilistic model on letter sequences [13]. The methods resemble each other. Both methods use the previous letter context to guess the next letter, but in the prefix-based approach, an incorrectly guessed letter is corrected immediately after it has been entered. Conversely, when using a probabilistic model on letter sequences, the user first inputs all letters in the word and then scrolls through a list of suggestion words matching the input.

Our own method utilizes a similar probabilistic model on letter sequences as [13]. The novel aspects of our method are (1) utilizing a morphologically segmented training corpus in order to construct a probabilistic model of words as morph sequences and (2) using phonological constraints for filtering impossible

suggestions. To the best of our knowledge, this has not been tried before in the domain of text-entry. In the related domain of speech recognition, similar approaches have yielded good results [2] for agglutinative languages.

3 A Probabilistic Model of Word Structure

Predictive text entry can be seen as a labeling task, where every key in a sequence of keys is assigned its most likely letter. The usual approach to such tasks is using stochastic models with hidden variables e.g. HMMs.

Though predictive text entry can be implemented fairly well using n-gram models (such as HMMs) on letter sequences, as exemplified by [13], there are problems with this approach. An HMM cannot encode very long dependencies inside words, which leads to difficulties since it is not possible to adequately separate stems from affixes or to handle long phonological dependencies like vowel harmony. Higher order HMMs are not useful in practice because of efficiency problems [13].

In order to construct a general prediction model, which still represents word structure at a higher level than at the level of single letters, we represent words as morph sequences, which are extracted from an automatically segmented training corpus.

To illustrate the usefulness of our approach we look at some Finnish word forms. Consider the word form "taloa" (sg. partitive case of the word house). Automatic segmentation of the training corpus might give the segmentation "talo+a", into the stem "talo" and the ending "a". If the word form "taloakin" (sg. partitive case of "talo" with the clitic "kin") does not occur in the training data, we can still estimate its probability by utilizing the frequencies of the morph combinations "talo + a" and "a + kin",

Our model for word structure is a Markov chain of morph sequences. Data sparseness is likely to be a serious problem, since there are tens of thousands of morphs, many of which only occur once. We therefore combine the Markov chain with an HMM which maps key sequences to letter sequences. The HMM does not utilize morph boundaries, so it gives some estimate for the probability of a word form like "a-l-a-t-a-l-o" (a common Finnish surname), even though the combination of morphs "ala + talo" would never have been observed in the training data and the morph sequence model would therefore be unable to give a good estimate for the probability of the compound word.

Finally many agglutinative languages like Finnish and Turkish incorporate phonological phenomena, such as vowel harmony, which can span over arbitrarily long distances in word forms. These phenomena cannot be adequately handled using n-gram models of morphs or letters, which has prompted us to include phonological constraints in our system.

The statistical models and phonological rules are implemented as weighted finite-state transducers, which allows us to combine them using the algebraic operations for finite-state transducers. Transducers are a natural choice for coding arbitrarily long dependencies such as vowel harmony.

3.1 A Hidden Markov Model for Predicting Letter Sequences from Key Sequences

We denote a sequence of mobile phone keys k_i of length n by $K = (k_i)_{i=1}^n$. Correspondingly, we denote a sequence of letters l_i of length n by $L = (l_i)_{i=1}^n$. For key k_i, we denote the corresponding set of letters by $\mathrm{M}(k_i)$. E.g. $\mathrm{M}(2) = \{\text{a, b, c, ä, å}\}$ on a typical Finnish mobile phone keyboard. For key sequence K, we denote the set of corresponding letter sequences by $\mathrm{M}(K)$.

The task of the letter model is to give the probability of a letter sequence L given a sequence of keys K. Naturally $\mathrm{P}(L|K) > 0$, iff $L \in \mathrm{M}(K)$. We give the standard third order HMM approximation for $\mathrm{P}(L|K)$ in equation (1). The second equality follows by noting that $\mathrm{P}(k_i|l_i) = 1$ for all i, since every letter corresponds to exactly one key. This effectively makes our HMM equivalent to a Markov chain. Three special letters l_{-2}, l_{-1}, l_0 are required to make the approximation work. These buffer symbols are added both to the training data and the suggestions. To counteract data sparseness, we smooth probabilities using lower order HMMs as explained in the following subsection.

$$\mathrm{P}(L|K) = \prod_{i=1}^{n} \mathrm{P}(k_i|l_i)\mathrm{P}(l_i|l_{i-3}, l_{i-2}, l_{i-1}) = \prod_{i=1}^{n} \mathrm{P}(l_i|l_{i-3}, l_{i-2}, l_{i-1}) \qquad (1)$$

3.2 A Markov Chain of Morphs

A morph of n letters in the training data is simply a sequence of n letters, so we denote it by $L = (l_i)_{i=1}^n$. A key sequence $K = (k_i)_{i=1}^m$ corresponds to a sequence of morphs $L_1 ... L_s$, where each $L_j = (l_{j_i})_{i=1}^{n_j}$, iff $\Sigma_{j=1}^{s} n_j = m$ and $l_{j_i} \in \mathrm{M}(k_{n_1+...+n_{j-1}+i})$ for all l_{j_i}. We denote the set of morph sequences that correspond to a key sequence K by $\mathcal{M}(K)$.

The task of the morph model is to assign a probability for each sequence of morphs in $\mathcal{M}(K)$ for the key sequence K. The probability of a sequence of s morphs $(L_1, ..., L_s) \in \mathcal{M}(K)$ is given by the chain rule of probabilities in equation (2).

$$\mathrm{P}(L_1, ..., L_s) = \mathrm{P}(L_1)\mathrm{P}(L_2|L_1) \ ... \ P(L_s|L_1, ..., L_{s-1}) \qquad (2)$$

We make the standard assumptions for a first order Markov model, namely that $\mathrm{P}(L_i|L_1, ..., L_{i-1}) = \mathrm{P}(L_i|L_{i-1})$, which means that we assume that the probability of a morph occurring depends only on its left neighboring morph and the morph itself. Thus we can approximate equation (2) by equation (3).

$$\mathrm{P}\big(L_1, ..., L_s\big) = \mathrm{P}(L_1)\mathrm{P}\big(L_2|L_1\big)\mathrm{P}\big(L_3|L_2\big) \ ... \ \mathrm{P}(L_s|L_{s-1}) \qquad (3)$$

In practice we use a training corpus for estimating the probability $\mathrm{P}(L_i|L_{i-1})$. For the morphs L_i and L_{i-1} we use the estimate in equation (4), where $\mathrm{C}(L_{i-1}, L_i)$ is the number of times the morph L_i followed the morph L_{i-1} in the training corpus and $\mathrm{C}(L_{i-1})$ is the count of the morph L_{i-1} in the training corpus.

$$\hat{\mathrm{P}}(L_i|L_{i-1}) = \mathrm{C}(L_{i-1}, L_i)/\mathrm{C}(L_{i-1}) \qquad (4)$$

Since many morphs L_i and L_{i-1} do not occur adjacently anywhere in the training corpus, we also utilize the unigram estimates $\hat{\mathrm{P}}(L_i) = \mathrm{C}(L_i)/S$ when estimating the probabilities $\mathrm{P}(L_i|L_{i-1})$. Here S is the size of the training corpus. The actual estimate for the probability $\mathrm{P}(L_i|L_{i-1})$ is given in equation (5). The coefficient a is determined by deleted interpolation (see [1]).

$$\mathrm{P}(L_i|L_{i-1}) = \hat{\mathrm{P}}(L_i|L_{i-1})^a \hat{\mathrm{P}}(L_i)^{1-a}, \text{ where } 0 \leq a \leq 1 \ . \tag{5}$$

3.3 Phonological Constraints

We use phonological constraints to filter the results given by the statistical components of the system. The result given by the system is thus the most probable string, which satisfies the phonological constraints. Formally they are two-level constraints, which can be implemented using the two-level compiler hfst-twolc[2].

3.4 Combining Models using Weighted Finite-State Calculus

Both the HMM on letter sequences and the morph sequence model are implemented as sets of weighted finite-state transducers. The models are compiled using the POS tagger tools, [12], in the hfst-interface[3]. We simply replace words and tags by keys, letters and morphs.

The input key sequence entered by the user is compiled into a finite state transducer, which codes all possible realizations of the key sequence as letter sequences. The realizations are weighted using the HMM model on letter sequences and the weighted letter sequences are coded into morph sequences. These morph sequences are then re-scored using the morph sequence model. Finally those morph sequences which do not satisfy the phonological constraints are filtered out.

In a last processing step, the morpheme boundaries are removed and the ten most likely letter sequences are extracted.

4 Data and Linguistic Resources

We trained predictive text entry systems for Finnish and Turkish to evaluate our method. We compare our results with two existing text entry systems by [11] and [13]. There are no standardized test materials for predictive text entry for Finnish or Turkish, but we were able to obtain the training materials and test materials used in the previous systems.

The training materials and test materials for both Finnish and Turkish were processed in the same way. All uppercase letters were transformed into lowercase letters and all words that included non-alphabetical characters were removed. This included among other characters such as numbers and punctuation except apostrophes in Turkish, which are used to signify the boundary between the stem and affix in some word forms.

[2] https://kitwiki.csc.fi/twiki/bin/view/KitWiki/HfstTwolC
[3] http://hfst.sf.net

4.1 Finnish

For training and testing the Finnish text entry system, we use the same data as [11], though in addition to the training data they use a morphological analyzer, which we do not utilize. The training material is extracted from Finnish IRC logs and contains some 350,000 words. The test material consists of $6,663$ words of actual text message data[4].

Phonological Constraints for Finnish In Finnish a word form, which is not a compound word, cannot contain both back-vowels ("a", "o", "u") and front-vowels ("ä", "ö", "y"). We implemented two two-level rules [6], which realize this constraint on a morphologically segmented word form.

Figure 2 shows one of the rules. The rule disallows an affix with front-vowels, together with a stem with back-vowels. The named regular expressions `Affix` and `FrontVowelAffix` are sets of know inflective and derivational affixes in Finnish. The expression `BackVowelStem` denotes sequences of four or more characters, where all vowels are back-vowels.

```
"Front Vowel Harmony"
  <[ FrontVowelAffix ]> /<== BackVowelStem Affix* _ ;
```

Fig. 2. Rule for Finnish front vowel harmony using the rule-syntax of hfst-twolc for rules whose center is a regular expression.

4.2 Turkish

For training and testing the Turkish text entry system, we use the same material as [13]. It is a corpus of news paper text containing some 20 million words. The material is divided into a test corpus containing $2,597$ words and a training corpus which includes the rest of the words in the material. Thus the training data and test data are disjoint.

With Turkish we do not use phonological constraints.

5 Evaluation

In this section, we present the results of experiments using the Finnish and Turkish training data and test data presented in the previous section. For Finnish we examine the impact of varying the amount of training data on the performance of the predictive text entry system. For Turkish we present results on the whole training material.

[4] The original test data contains $10,851$ words, but it turned out that the latter part of the test data file is actually a uniquified list of words, which skews test results, so we decided to only use the earlier half of the material

5.1 The Keystrokes Per Characters Ratio

In this paper we use the keystrokes per character (KPC) ratio for measuring the efficiency of text entry. The KPC ratio for a text entry method is computed as the average number of keystrokes required to input one letter in a test corpus. Following [13], we do not consider space characters as a part of the test data.

By examining the schematic picture of a mobile phone keypad in Figure 1, it can be seen that the key sequence needed to input the word "kukka" (flower) on a mobile phone with Finnish keypad and using the multitap input method is `5-5-8-8-5-5-<NEXT>-5-5-2`. The `<NEXT>`-key is required after entering the first "k" in order to tell the text entry that the next press of key `5` starts a new symbol. This increases the number of keystrokes from 9 to 10. On test data consisting solely of the word "kukka", the KPC ratio would thus be $10/5 = 2.0$.

When computing the KPC ratio for predictive text entry methods, we assume that multitap is used as a fallback method when entering OOV words, i.e. words that are not found among the suggestions given by the system. In detail, entering an OOV word requires:

1. Entering the keys for the letters used to write the word (one keystroke per letter).
2. Scrolling through the suggestions (9 keystrokes in our system, since 10 suggestions are given).
3. Deleting the last suggestion one letter at a time using a backspace key (one keystroke per letter). [5]
4. Switching to multitap mode using a special key (one keystroke).
5. Inputting the word in multitap mode (keystroke count varies depending on the word).
6. Switching back to predictive mode using a special key (one keystroke).

5.2 Results for Finnish

We constructed 12 text entry systems using different portions of the training data for Finnish presented in Section 4. We used the first $1,000$, $35,000$, $69,000$, $103,000$, $137,000$, $171,000$, $205,000$, $239,000$, $273,000$, $307,000$, $341,000$ and $345,337$ words respectively. The impact of the size of the training data is shown in Figure 3. The minimum KPC ratio 1.3748 was attained for the entire training data consisting of $345,337$ words.

We also evaluated the effect of the different components on the KPC of the predictive text entry system. The results are shown in Table 1.

We compared our system to another published Finnish text-entry system by [11], which is based on a colloquial dictionary compiled from running text and a morphological analyzer. The authors do not evaluate their system using the KPC ratio, but we were able to obtain their test results and according to our

[5] Many commercial phones make it possible to delete an entire word using one keystroke. However, deleting the word one letter at a time is consistent with the evaluation procedure used in [13].

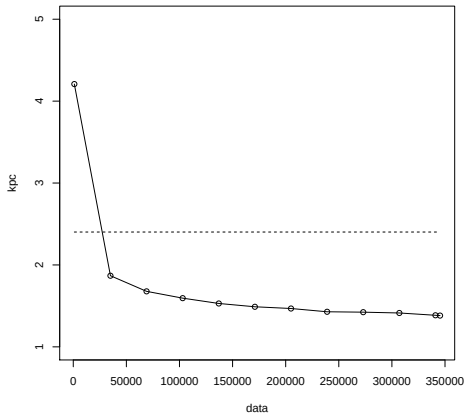

Fig. 3. The effect of the amount of training data (horizontal axis) on the KPC ratio (vertical axis) of the Finnish predictive text entry system. The dashed line marks the KPC for multitap 2.4018. The minimal KPC ratio is 1.3738.

experiments they achieve a KPC ratio of 1.6120. This means that our system achieves a 15.1% point decrease in KPC compared with their system using the same training materials and test data, but without using the morphological analyzer[6].

Table 1. KPC for Finnish Multitap and for using different components of our system. The third column shows the improvement over multitap.

Method	KPC	Improvement (%)
Multitap	2.4018	0.0
Letter n-grams	1.7368	27.7
Letter n-grams and morph sequence model	1.3751	42.7
Letter n-grams, morph sequence model and rules	1.3748	42.8

In practice, ten suggestions for an input sequence is quite a lot. Few users are likely to scroll through ten suggestions especially if there are many non-words in the suggestion list. Therefore we also computed the KPC ratio for the entire

[6] When examining the test data used by [11], we discovered, that the latter half of the data consisted of a uniquified word list, which affected their results negatively. We have computed the KPC ratio for both our own system and the system of [11] using only the 6,663 first words in the test data.

training data as a function of the number of suggestions given by the system. The results are shown in Table 2.

Table 2. The effect of the number of suggestions on the KPC ratio of the Finnish text entry system using all of the training data.

# of Sugg.	1	2	3	4	5	6	7	8	9	10
KPC	2.1478	1.7363	1.5872	1.5153	1.4602	1.4257	1.3998	1.3849	1.3798	1.3748

Finally we wanted to compare our system to some commercially available text entry systems. To accomplish this, we took a list of thirty words chosen at random from the test data, entered the words into three commercially available mobile phones and computed the KPC ratio. We also tested our own system using the words. Since the text entry system of the mobile phones did not give ten suggestions, we computed results only on the three first guesses given by each system. The results are shown in Table 3.

Table 3. The KPC ratio for a 30 word random sample from our test data using the multitap method, three commercial mobile phones and our text entry system. Only the first three suggestions for each input sequence were considered for the predictive text entry systems.

System	KPC
Multitap	2.3
Nokia C7	2.2
Nokia 2600	2.0
Samsung SGH M310	2.0
Our system	1.4

5.3 Results for Turkish

For Turkish, we trained one system using the entire 20 million word training corpus, which was presented in Section 4. We compare our results against the predictive text entry system by [13].[7] The results are shown in Table 4.

[7] For Turkish our computation gives the KPC ratio 2.4386 for the multitap method. This differs slightly from the figure 2.2014 given by [13]. Thus it is possible that our results are not entirely comparable to the results of [13]. The improvement in KPC ratio given for the method by [13] in Table 4 is computed using our figure for the KPC ratio of the multitap method. The improvement given by [13] is 35%.

Table 4. KPC for Turkish using different input methods. The third column shows the improvement over the multitap method.

Method	KPC	Improvement (%)
Multitap	2.4386	0.0
Letter n-grams [13]	1.4382	41.0
Our method	1.1800	51.6

6 Discussion

For Finnish, our system achieves a substantial 15.1% point drop in KPC ratio compared with the other system which utilizes a morphological analyzer. In their Finnish text entry system [11] give only 3 suggestions. Looking at Table 2 we see that the KPC ratio for our system is 1.5872, when only considering the three first suggestions, which is still lower than the KPC ratio 1.6120, which their system achieves. Considering, that except the phonological constraints, we use only language independent components, this is remarkable.

The phonological constraints seem to be having very little effect, as can be seen in Table 1. The decrease in KPC when using the phonological constraints is only about 0.1%. This may be a result of the unsupervised segmentation, which does not always succeed in finding the correct morpheme boundaries and therefore may prevent the rules from being triggered.

As Table 3 shows, our system outperforms three commercially available mobile phones on a thirty word test set chosen at random from our test data. This shows that our approach has great practical potential.

As can be seen in Table 4 our method achieves an additional 10% point reduction in KPC for Turkish compared with the system in [13]. Our KPC ratio 1.1800 needs to be related to the fact that our method can never achieve a KPC ratio lower than 1, since every letter in a word needs to be typed. We achieve a 52% reduction in KPC for the Turkish test data compared with the multitap method. A fast computation reveals that the maximal possible reduction is only 59%, which demonstrates that our system is nearly optimal on the Turkish data.

7 Conclusions and future work

We have demonstrated a highly accurate predictive text entry model, which can be constructed using unsupervised methods. Additionally linguistic rules can be added to improve the performance of the system.

There are several interesting future research directions. In order to reduce the KPC ratio to < 1, the system should be able to predict morphs before they are completely typed. We should also consider adding a model, which extends over word boundaries. Further, it would be interesting to examine the effect of the segmentation of the training corpus on the function of the phonological rules. A linguistically soundly segmented training corpus would probably allow the rules to act more often and thus improve the KPC ratio.

8 Acknowledgments

We wish to thank Cüneyd Tantuğ for the Turkish data, Sam Hardwick for the Finnish training data and Jarmo Wideman for drawing Figure 1. The first author is funded by the graduate school Langnet. Finally we wish to thank the anonymous reviewers for their valuable work.

References

1. Brants, T.: Tnt - a statistical part-of-speech tagger. In: Proceedings of the Sixth Applied Natural Language Processing. pp. 224–231. ACL, Seattle (2000)
2. Creutz, M., Hirsimäki, T., Kurimo, M., Puurula, A., Pylkkönen, J., Siivola, V., Varjokallio, M., Arisoy, E., Saraçlar, M., Stolcke, A.: Morph-based speech recognition and modeling of out-of-vocabulary words across languages. ACM Transactions on Speech and Language Processing 5(1) (2009)
3. Creutz, M., Lagus, K.: Unsupervised models for morpheme segmentation and morphology learning. ACM Transactions on Speech and Language Processing 4(1) (2007)
4. Dale Grover, Martin King, C.K.: Reduced keyboard disambiguating computer. Patent US 5818437 (1998)
5. Klarlund, N.: Word n-grams for cluster keyboards. In: TextEntry '03 Proceedings of the 2003 EACL Workshop on Language Modeling for Text Entry Methods. pp. 51–58. ACL, Stroudsburg (2003)
6. Koskenniemi, K.: Two-level Morphology: A General Computational Model for Word-Form Recognition and Production. Ph.D. thesis, University of Helsinki (1983)
7. Lindén, K., Axelson, E., Hardwick, S., Silfverberg, M., Pirinen, T.: HFST–framework for compiling and applying morphologies. In: Mahlow, C., Piotrowski, M. (eds.) Systems and Frameworks for Computational Morphology SFCM 2011. CCIS, vol. 100, pp. 67–85. Springer, Heidelberg (2011)
8. MacKenzie, I.S.: KSPC (keystrokes per character) as a characteristic of text entry techniques. In: Paternò, F. (ed.) Human Computer Interaction with Mobile Devices HCI 2002. LNCS, vol. 2411, pp. 195–210. Springer, Heidelberg (2002)
9. Mackenzie, I.S., Kober, H., Smith, D., Jones, T., Skepner, E.: Letterwise: Prefix-based disambiguation for mobile text input. In: Proceedings of the 14th annual ACM Symposium on User Interface software and technology. pp. 111–120. ACM, Orlando (2001)
10. Scott MacKenzie, W.S.: Text entry for mobile computing: Models and methods, theory and practice. Human-Computer Interaction 17(2), 147–198 (2002)
11. Silfverberg, M., Hyvärinen, M., Pirinen, T.: Improving predictive entry of Finnish text messages using IRC logs. In: Jassem, K., Fuglewicz, P., Piasecki, M., Przepiórkowski, A. (eds.) Proceedings of the Computational Liguistics-Applications Conference. Jachranka (2011)
12. Silfverberg, M., Lindén, K.: Combining statistical models for POS tagging using finite-state calculus. In: Pedersen, B.S., Nešpore, G., Skadina, I. (eds.) 18th Nordic Conference on Computational Linguistics. pp. 183–190 (2011)
13. Tantuğ, A.C.: A probabilistic mobile text entry system for agglutinative languages. IEEE Transactions on Consumer Electronics 56(4), 1018–1024 (2010)